STEP_5_LLAMA_DATASET_API.md

Step 5: Llama Dataset API Extension - Complete Implementation Guide

Overview

Step 5 enhances the `/llama/examples` endpoint with real database integration, embeddings support, and comprehensive Ollama integration for RAG-style workflows.

Architecture

```
API Service (Port 8003)
├── /llama/examples endpoint
│   ├── Query keyword_hits (filtered by keyword/category)
│   ├── Join clips (file paths, timestamps)
│   ├── Join transcripts (context text)
│   ├── Optional: embeddings (if include_embeddings=true)
│   └── Format for Llama consumption
│
├── Database Integration
│   ├── keyword_hits table
│   ├── clips table
│   ├── transcripts table
│   └── embeddings table
│
└── Ollama Integration
├── Prompt engineering
├── Context assembly
└── Strategy generation
```

Implementation Tasks

Task 1: Database Integration for /llama/examples

Location: `services/api-service/app/main.py`

Replace the placeholder `/llama/examples` endpoint with full database implementation:

```
from sqlalchemy.orm import Session
from app.database import get_db
from app.models import KeywordHit, Clip, Transcript, Embedding, MediaItem
from fastapi import Depends
```

```
@app.get("/llama/examples", tags=["Llama Dataset"])
async def get_llama_examples(
keyword: Optional[str] = Query(None),
top_k: int = Query(5, ge=1, le=50),
include_embeddings: bool = Query(False),
category: Optional[str] = Query(None),
db: Session = Depends(get_db)
):
"""
Get structured training examples for Llama fine-tuning or prompting

Query Parameters:
- keyword: Filter by keyword (e.g., "RSI", "breakout")
- top_k: Number of examples to return
- include_embeddings: Whether to include embedding vectors
- category: Filter by keyword category (technical_indicator, price_action, etc.)

Returns:
- List of examples with transcripts, clips, detected concepts, and optional embed
"""
try:
    # Build query
    query = db.query(KeywordHit)

    if keyword:
        query = query.filter(KeywordHit.keyword.ilike(f"%{keyword}%"))

    if category:
        query = query.filter(KeywordHit.category == category)

    # Order by confidence and limit
    keyword_hits = query.order_by(KeywordHit.confidence.desc()).limit(top_k).al

    examples = []

    for hit in keyword_hits:
        # Get associated clip
        clip = db.query(Clip).filter(Clip.keyword_hit_id == hit.id).first()

        # Get transcript context (segments around the hit)
```

```
        transcript_segments = db.query(Transcript).filter(
            Transcript.media_item_id == hit.media_item_id,
            Transcript.start_time >= hit.start_time - 10,  # 10 seconds before
            Transcript.end_time <= hit.end_time + 10        # 10 seconds after
        ).order_by(Transcript.segment_index).all()

        transcript_text = " ".join([seg.text for seg in transcript_segments])

        # Build example
        example = {
            "clip_id": f"media_{hit.media_item_id}_{hit.keyword}_{hit.id}",
            "transcript": transcript_text,
            "keyword": hit.keyword,
            "category": hit.category if hasattr(hit, 'category') else None,
            "timestamp": hit.start_time,
            "clip_url": f"/clip/{clip.id}/download" if clip else None,
            "detected_concepts": [],  # TODO: Add from ML service
            "context_text": hit.context_text,
            "confidence": hit.confidence
        }

        # Add frame path if available
        if clip:
            example["frame_path"] = f"/data/processed/clips/media_{hit.media_item_
            example["start_time"] = clip.start_time
            example["end_time"] = clip.end_time

        # Add embeddings if requested
        if include_embeddings:
            embedding = db.query(Embedding).filter(
                Embedding.media_item_id == hit.media_item_id,
                Embedding.reference_id == hit.id,
                Embedding.embedding_type == 'keyword'
            ).first()

            if embedding:
                example["embeddings"] = embedding.embedding_vector
                example["embedding_model"] = embedding.embedding_model
```

```
            else:
                example["embeddings"] = None

        examples.append(example)

    return {
        "examples": examples,
        "total": len(examples),
        "keyword_filter": keyword,
        "category_filter": category
    }

except Exception as e:
    logger.error(f"Error getting Llama examples: {str(e)}")
    raise HTTPException(status_code=500, detail=str(e))
```

Task 2: Create Llama Integration Module

Create new file: services/api-service/app/llama_client.py

services/api-service/app/llama_client.py

Client for interacting with Ollama for strategy generation

```
import requests
import logging
from typing import List, Dict, Optional

logger = logging.getLogger(__name__)

class LlamaClient:
"""Client for Ollama API integration"""

    def __init__(self, base_url: str = "http://localhost:11434"):
        self.base_url = base_url
        self.api_url = f"{base_url}/api/generate"

    def generate_strategy_from_examples(
        self,
```

```
    examples: List[Dict],
    model: str = "llama2",
    temperature: float = 0.7
) -> Dict:
    """
    Generate trading strategy from Llama examples

    Args:
        examples: List of keyword examples with transcripts
        model: Ollama model name
        temperature: Generation temperature (0-1)

    Returns:
        Dict with generated strategy and metadata
    """
    try:
        # Build context from examples
        context = self._build_context(examples)

        # Build prompt
        prompt = self._build_strategy_prompt(context, examples)

        # Call Ollama
        response = requests.post(
            self.api_url,
            json={
                "model": model,
                "prompt": prompt,
                "stream": False,
                "options": {
                    "temperature": temperature,
                    "num_predict": 500
                }
            },
            timeout=60
        )

        if response.status_code != 200:
```

```
            logger.error(f"Ollama error: {response.text}")
            return {"status": "failed", "error": response.text}

        result = response.json()

        return {
            "status": "success",
            "strategy": result.get("response", ""),
            "model": model,
            "context_examples": len(examples),
            "prompt_length": len(prompt)
        }

    except Exception as e:
        logger.error(f"Error generating strategy: {str(e)}")
        return {"status": "failed", "error": str(e)}

def _build_context(self, examples: List[Dict]) -> str:
    """Build context string from examples"""
    context_parts = []

    for i, ex in enumerate(examples, 1):
        context_parts.append(
            f"{i}. {ex['keyword']} at {ex['timestamp']}s:\n"
            f"   {ex['transcript'][:200]}...\n"
        )

    return "\n".join(context_parts)

def _build_strategy_prompt(self, context: str, examples: List[Dict]) -> str:
    """Build strategy generation prompt"""
    keywords = list(set([ex['keyword'] for ex in examples]))

    prompt = f"""You are a trading strategy expert. Based on the following video
```

Video Excerpts:
{context}

Key Concepts: {', '.join(keywords)}

Please provide:

1. Strategy Overview (2-3 sentences)
2. Entry Conditions (specific indicators and levels)
3. Exit Conditions (take profit and stop loss)
4. Risk Management (position sizing, max drawdown)
5. Timeframe (best timeframe for this strategy)

Be specific with indicator values and thresholds where mentioned in the excerpts."""

```
    return prompt

def summarize_concept(
    self,
    keyword: str,
    examples: List[Dict],
    model: str = "llama2"
) -> str:
    """
    Summarize a trading concept from examples

    Args:
        keyword: The trading keyword/concept
        examples: Examples containing the keyword
        model: Ollama model name

    Returns:
        Summary text
    """
    try:
        context = self._build_context(examples)

        prompt = f"""Based on these trading video excerpts about {keyword}, prov
```

Excerpts:
{context}

Explanation:"""

```
        response = requests.post(
            self.api_url,
```

```
            json={
                "model": model,
                "prompt": prompt,
                "stream": False,
                "options": {"temperature": 0.5, "num_predict": 150}
            },
            timeout=30
        )

        if response.status_code == 200:
            return response.json().get("response", "")
        else:
            return f"Error: {response.text}"

    except Exception as e:
        logger.error(f"Error summarizing concept: {str(e)}")
        return f"Error: {str(e)}"
```

Task 3: Add Strategy Generation Endpoint

Add to: services/api-service/app/main.py

from app.llama_client import LlamaClient

Initialize Llama client on startup

llama_client: Optional[LlamaClient] = None

@app.on_event("startup")
async def startup_event():
"""Initialize clients on startup"""
global llama_client
try:
ollama_url = os.getenv("OLLAMA_URL", "http://localhost:11434")
llama_client = LlamaClient(base_url=ollama_url)
logger.info("Llama client initialized")
except Exception as e:
logger.error(f"Error initializing Llama client: {str(e)}")

@app.post("/llama/generate-strategy", tags=["Llama Dataset"])
async def generate_strategy_from_keyword(
keyword: str,
top_k: int = Query(5, ge=1, le=20),
model: str = Query("llama2"),
temperature: float = Query(0.7, ge=0.0, le=1.0),

db: Session = Depends(get_db)
):
"""
Generate a trading strategy using Llama based on keyword examples

Query Parameters:
- keyword: Trading keyword to base strategy on (e.g., "RSI", "MACD")
- top_k: Number of examples to use as context
- model: Ollama model name (default: llama2)
- temperature: Generation temperature (0-1)

Returns:
- Generated strategy with entry/exit conditions and risk management
"""
try:
    if llama_client is None:
        raise HTTPException(status_code=503, detail="Llama client not initialized")

    # Get examples
    examples_response = await get_llama_examples(
        keyword=keyword,
        top_k=top_k,
        include_embeddings=False,
        category=None,
        db=db
)

    examples = examples_response["examples"]

    if not examples:
        raise HTTPException(status_code=404, detail=f"No examples found for key

    # Generate strategy
    result = llama_client.generate_strategy_from_examples(
        examples=examples,
        model=model,
        temperature=temperature
)

```
    if result["status"] == "failed":
        raise HTTPException(status_code=500, detail=result["error"])

    return {
        "keyword": keyword,
        "strategy": result["strategy"],
        "model": result["model"],
        "examples_used": result["context_examples"],
        "timestamp": datetime.utcnow().isoformat()
    }

except HTTPException:
    raise
except Exception as e:
    logger.error(f"Error generating strategy: {str(e)}")
    raise HTTPException(status_code=500, detail=str(e))
```

@app.get("/llama/summarize/{keyword}", tags=["Llama Dataset"])
async def summarize_keyword_concept(
keyword: str,
top_k: int = Query(3, ge=1, le=10),
db: Session = Depends(get_db)
):
"""
Get a Llama-generated summary of a trading concept

```
Path Parameters:
- keyword: Trading keyword to summarize

Query Parameters:
- top_k: Number of examples to base summary on

Returns:
- Concise explanation of the concept
"""
try:
    if llama_client is None:
        raise HTTPException(status_code=503, detail="Llama client not initialized")
```

```
    # Get examples
    examples_response = await get_llama_examples(
        keyword=keyword,
        top_k=top_k,
        include_embeddings=False,
        category=None,
        db=db
    )

    examples = examples_response["examples"]

    if not examples:
        raise HTTPException(status_code=404, detail=f"No examples found for key

    # Generate summary
    summary = llama_client.summarize_concept(
        keyword=keyword,
        examples=examples
    )

    return {
        "keyword": keyword,
        "summary": summary,
        "examples_used": len(examples),
        "timestamp": datetime.utcnow().isoformat()
    }

except HTTPException:
    raise
except Exception as e:
    logger.error(f"Error summarizing concept: {str(e)}")
    raise HTTPException(status_code=500, detail=str(e))
```

Task 4: Create Python Integration Examples

Create new file: `docs/LLAMA_INTEGRATION_EXAMPLES.md`

Llama Integration Examples

Python Script for Strategy Generation

```
import requests
import json

# Configuration
API_BASE = "http://localhost:8003"
KEYWORD = "RSI"

# 1. Get examples for the keyword
examples_response = requests.get(
f"{API_BASE}/llama/examples",
params={
"keyword": KEYWORD,
"top_k": 5,
"include_embeddings": False
}
)

examples = examples_response.json()
print(f"Found {examples['total']} examples for '{KEYWORD}'")

# 2. Generate strategy using Llama
strategy_response = requests.post(
f"{API_BASE}/llama/generate-strategy",
params={
"keyword": KEYWORD,
"top_k": 5,
"model": "llama2",
"temperature": 0.7
}
)

strategy = strategy_response.json()
print("\n" + "="*60)
print(f"TRADING STRATEGY FOR {KEYWORD}")
print("="*60)
print(strategy['strategy'])
print("="*60)
```

3. Get concept summary

summary_response = requests.get(
f"{API_BASE}/llama/summarize/{KEYWORD}",
params={"top_k": 3}
)

summary = summary_response.json()
print(f"\nConcept Summary: {summary['summary']}")

Direct Ollama Integration

import requests

Get examples from API

examples_response = requests.get(
"http://localhost:8003/llama/examples",
params={"keyword": "MACD", "top_k": 5}
)

examples = examples_response.json()["examples"]

Build custom prompt

context = "\n".join([
f"- {ex['transcript'][:150]}... (at {ex['timestamp']}s)"
for ex in examples
])

prompt = f"""Based on these trading video excerpts about MACD:

{context}

Generate a step-by-step trading strategy using MACD. Include:

1. Entry signals
2. Exit signals
3. Stop loss placement
4. Position sizing"""

Send to Ollama directly

ollama_response = requests.post(
"http://localhost:11434/api/generate",
json={
"model": "llama2",
"prompt": prompt,
"stream": False

```
}
)

strategy = ollama_response.json()["response"]
print(strategy)
```

Batch Processing Multiple Keywords

```
import requests
from concurrent.futures import ThreadPoolExecutor

API_BASE = "http://localhost:8003"
KEYWORDS = ["RSI", "MACD", "Bollinger Bands", "Support", "Resistance"]

def generate_strategy(keyword):
"""Generate strategy for a keyword"""
response = requests.post(
f"{API_BASE}/llama/generate-strategy",
params={"keyword": keyword, "top_k": 5}
)
return keyword, response.json()
```

Process in parallel

```
with ThreadPoolExecutor(max_workers=3) as executor:
results = executor.map(generate_strategy, KEYWORDS)

for keyword, result in results:
print(f"\n{'='*60}")
print(f"STRATEGY: {keyword}")
print('='*60)
print(result['strategy'])
```

Task 5: Update API Documentation

Update: docs/API_DOCUMENTATION.md

Add new section after the existing Llama Dataset API section:

Llama Strategy Generation (NEW)

POST /llama/generate-strategy

Generate a complete trading strategy using Llama based on keyword examples

Query Parameters:

- keyword (string, required): Trading keyword (e.g., "RSI", "breakout")
- top_k (int, default: 5): Number of examples to use as context
- model (string, default: "llama2"): Ollama model name
- temperature (float, default: 0.7): Generation temperature (0-1)

Example:
curl -X POST "http://localhost:8003/llama/generate-strategy?keyword=RSI&top_k=5&model=llama2&temperature=0.7"

Response:
{
"keyword": "RSI",
"strategy": "Strategy Overview:\nThis RSI-based momentum strategy...\n\nEntry Conditions:\n1. RSI crosses above 30 (oversold)\n2. Price confirms with bullish candle...",
"model": "llama2",
"examples_used": 5,
"timestamp": "2025-11-20T21:00:00Z"
}

GET /llama/summarize/{keyword}

Get a concise Llama-generated summary of a trading concept

Path Parameters:

- keyword (string): Trading keyword to summarize

Query Parameters:

- top_k (int, default: 3): Number of examples to base summary on

Example:
curl "http://localhost:8003/llama/summarize/MACD?top_k=3"

Response:
{
"keyword": "MACD",
"summary": "MACD (Moving Average Convergence Divergence) is a momentum indicator that shows the relationship between two moving averages. It generates buy signals when the MACD line crosses above the signal line and sell signals when it crosses below. Traders also watch for divergences between MACD and price action as potential reversal signals.",
"examples_used": 3,
"timestamp": "2025-11-20T21:00:00Z"
}

Testing Step 5

End-to-End Test Script

Create: tests/test_llama_integration.py

```
import requests
import time

API_BASE = "http://localhost:8003"

def test_llama_pipeline():
"""Test complete Llama integration pipeline"""
```

```
print("Step 1: Testing /llama/examples endpoint...")
examples_response = requests.get(
    f"{API_BASE}/llama/examples",
    params={"keyword": "RSI", "top_k": 3}
)
assert examples_response.status_code == 200
examples = examples_response.json()
print(f"✓ Found {examples['total']} examples")

print("\nStep 2: Testing /llama/generate-strategy endpoint...")
strategy_response = requests.post(
    f"{API_BASE}/llama/generate-strategy",
    params={"keyword": "RSI", "top_k": 3}
)
assert strategy_response.status_code == 200
strategy = strategy_response.json()
print(f"✓ Generated strategy ({len(strategy['strategy'])} chars)")

print("\nStep 3: Testing /llama/summarize endpoint...")
summary_response = requests.get(
    f"{API_BASE}/llama/summarize/RSI",
    params={"top_k": 2}
)
assert summary_response.status_code == 200
summary = summary_response.json()
print(f"✓ Generated summary: {summary['summary'][:100]}...")

print("\n✅ All Llama integration tests passed!")
```

if **name** == "**main**":
test_llama_pipeline()

Implementation Checklist

- [ ] Update /llama/examples endpoint with database integration
- [ ] Create llama_client.py module
- [ ] Add LlamaClient initialization in startup event
- [ ] Implement /llama/generate-strategy endpoint
- [ ] Implement /llama/summarize/{keyword} endpoint

- [ ] Create Python integration examples documentation
- [ ] Update API documentation with new endpoints
- [ ] Create end-to-end test script
- [ ] Test with sample data
- [ ] Verify Ollama connectivity

Configuration

Update `.env` file:

```
# Ollama Configuration
OLLAMA_URL=http://ollama:11434
OLLAMA_MODEL=llama2
OLLAMA_TEMPERATURE=0.7
```

Next Steps

After Step 5 completion:

Step 6: Strategy & Backtesting Framework

- Feature engineering from embeddings
- ML/RL model training
- Enhanced backtesting service
- Strategy promotion logic

Step 7: Tests & Acceptance

- Unit tests for all endpoints
- Integration tests
- Performance benchmarks
- CI/CD pipeline

Status

✅ **Step 5 Ready for Implementation** - Llama Dataset API Extension

- Database integration patterns defined
- LlamaClient module designed
- 3 new endpoints specified
- Integration examples created
- Testing approach documented

Implementation Time Estimate: 2-3 hours